REMARKS

The Office Action mailed October 5, 2005 has been carefully considered. Reconsideration in view of the following remarks is respectfully requested.

In view of the Examiner's earlier restriction requirement, Applicant retains the right to present claims 9-12 and 14-24 in a divisional Application.

With this amendment it is respectfully submitted the claims satisfy the statutory requirements.

The 35 U.S.C. § 112, Second Paragraph Rejection:

Claims 1-8, 13, 25 and 26 were rejected under 35 U.S.C. § 112, second paragraph, as allegedly being indefinite for failing to particularly point out and distinctly claim the subject matter applicant regards as the invention. This objection is respectfully traversed.

The Office Action takes exception to the use of the term "run-test/idle state" in the claims, and argues that it is unclear "if it is a run-test state and a idle state or a run-test state or a idle state." *Office Action, page 3.* The answer, however, is neither. "Run-test/idle" is the name of a single state. This is a commonly used term in the art and is used consistently in the specification and figures to refer to a single state. While Applicant realizes that the use of the "/" symbol may have led to this misunderstanding, Applicant is unsure on how to make the term any clearer as the "/" is included in the name of the state as is commonly known in the art, and is also used the same way in the specification. Therefore, Applicant is unaware of a way to modify the

claims without introducing a new name for the state that does not have support in the specification or art. For the Examiner's reference, Applicant attaches an exemplary reference entitled "Using the Internet to Repair Hardware in the Field" that utilizes this term on page 78 in reference to an IEEE 1532 or 1149.1-compliant device.

The 35 U.S.C. § 103 Rejection:

Claims 1-8, 13, 25 and 26 were rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Gruetzner et al.[1], in view of Parker et al.[2], among which claims 1, 5, 13, 25 and 26 are independent claims. This rejection is respectfully traversed.

> According to the Manual of Patent Examining Procedure (M.P.E.P.),
>
> To establish a *prima facie* case of obviousness, three basic criteria must be met. First there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in the applicant's disclosure.[3]

[1] U.S. Patent No. 5,444,715
[2] U.S. Patent No. 5,513,188
[3] M.P.E.P § 2143.

The Office Action alleges that "Applicants further point to the specification of the present application to elaborate on the definition of AC coupled interconnection. The Examiner would like to point out that a teaching reference (not used in the rejection), Whetsel teaches in Figures 1 and 2 a DC and AC interconnect that are connected by a capacitor. Although the prior art of record, Gruetzner, does not explicitly state the word capacitor as pointed out by the Applicants, it is well understood in the art, as shown by Whetsel, that AC interconnects are capacitively coupled." Applicant respectfully disagrees.

Whetsel shows an AC interconnect that is capacitively coupled in Figure 2. The Patent Office seems to interpret this as indicating that **all** AC interconnects are capacitively coupled. This is not correct, however. AC interconnections can and are made without the use of a capacitor. Whetsel simply happens to depict an AC interconnection that is capacitively coupled. It does not, however, follow from Whetsel that all AC interconnects are capacitively coupled. Nothing in Whetsel teaches or suggest this.

As such, Whetsel is irrelevant in interpreting what "AC interconnect" means in Gruetzner. As the Office Action pointed out, Gruetzner is silent as to the use of capacitors in AC interconnects. Nothing in Whetsel implies that invention is Gruetzner must use capacitors in its AC interconnects.

Furthermore, other aspects of Gruetsner seem to suggest that the AC interconnects in Gruetsner in fact do NOT utilize capacitors. Specifically, it appears that Gruetzner would be inoperative if used with AC coupled interconnects (AC interconnects with capacitors). This is

because the capturing of test data is at the receiver's end. Its likely that the receiver samples "logic level" of incoming signals with a system clock. As such, the test would fail if the interconnect has a small Resistance-Capacitance (RC) time constant. If the AC interconnect in Gruetsner included a capacitor, the transition pulse after the capacitor would be shorter than the hold time of the capture clock for the receiver, causing the test to fail. As such, it is extremely unlikely that the AC interconnects described in Gruetsner include capacitors. Therefore, Applicant respectfully maintains that Gruetsner fails to teach an AC coupled interconnect as stated in claim 1.

As to independent claims 13, 25, and 26, these claims also discuss AC coupled interconnects, and are therefore allowable for the same reasons as claim 1.

As to dependent claims 2-4 and 6-8, the argument set forth above is equally applicable here. The base claims being allowable, the dependent claims must also be allowable.

In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance.

Conclusion

It is believed that this Amendment places the above-identified patent application into condition for allowance. Early favorable consideration of this Amendment is earnestly solicited.

If, in the opinion of the Examiner, an interview would expedite the prosecution of this application, the Examiner is invited to call the undersigned attorney at the number indicated below.

Applicant respectfully requests that a timely Notice of Allowance be issued in this case. Please charge any additional required fee or credit any overpayment not otherwise paid or credited to our deposit account No. 50-1698.

Respectfully submitted,
THELEN REID & PRIEST, L.L.P.

Dated: 1/5/06

Marc S. Hanish
Reg. No. 42,626

Thelen Reid & Priest LLP
P.O. Box 640640
San Jose, CA 95164-0640
Tel. (408) 292-5800
Fax. (408) 287-8040

Using the Internet to repair hardware in the field

Neil G. Jacobson
Xilinx, Inc.

Abstract
The advent and recent approval of IEEE Std 1532 has provided the foundation for major improvements in both the quality and the manner in which programmable devices and systems can be accessed for test and configuration. This paper describes how reconfigurable systems comprised of IEEE Std 1532 compliant devices can be tested and repaired via the Internet (or any network).

Introduction
The advent and recent approval of IEEE Std 1532 [2] has provided the foundation for major improvements in both the quality and the manner in which programmable devices and systems can be tested. IEEE Std 1532 extends and enhances IEEE Std 1149.1 (commonly known as the boundary-scan standard) [1] to describe fully and document the operation and external behavior of programmable logic devices. This paper will discuss the development and details of IEEE Std 1532. Then it will describe how IEEE Std 1532 can be used to configure a single programmable device. Finally, this paper will discuss the manner in which this turnkey configurability can be exploited to develop reconfigurable systems that can be both tested and repaired remotely.

IEEE Std1532
The development of IEEE Std 1532 began with a preliminary meeting held April 19, 1996. There, a group of Programmable Device vendors and users discussed the possibility of standardizing the programming process for these devices. Subsequent meetings were held regularly and agreement was developed for basing the programming process upon the IEEE Std 1149.1 communication protocol, because many of these devices were expected to support the IEEE Std 1149.1 standard for testing purposes. This group adopted as its mission:

To define, document and promote the use of a standardized process and methodology for implementing programming capabilities within programmable integrated circuit devices, utilizing (and compatible with) the IEEE Std 1149.1 communication protocol. This standard would allow the programming of one or more compliant devices concurrently, while mounted on a board or embedded in a system, known as "In-System Configuration." Concurrent programming may often result in significant programming time efficiencies. The In-System feature would address the need to configure or reconfigure, read back, verify or erase programmable devices after they have been installed by a manufacturing process. This eliminates handling damage and the need for manufacturing steps and inventory management related to preprogrammed devices.

This work received the support of the IEEE Std 1149.1 Working Group which ultimately urged that this effort become its own standard within the Test Technology Technical Committee of the IEEE Computer Society. IEEE Std 1532 was then formally initiated in July of 1998.

The standard describes a series of mandatory and optional programming instructions and associated data registers that define a standard methodology for accessing and configuring programmable devices. These additional registers and instructions extend the capabilities of devices that comply with IEEE Std 1149.1 such that the 1149.1 Test Access Port may be used for configuration activities. A data description format and extensions to Boundary-Scan Description Language (BSDL) are also specified that provide for the development of standardized automation tools for device programming. A typical printed circuit board may contain many types of devices. Of these, some could be compliant with IEEE Std 1149.1 for the support of testing activities. Some devices on such a board might be In-System Configurable, meaning they are programmable devices that may not have their programming states loaded into them before the board has been assembled. These devices may be programmed after they have been attached to the board, via a physical and logical protocol. The physical protocol for programming utilizes the IEEE Std 1149.1 Test Access Port (TAP) pins, which allows a multiplicity of devices (both ISC and conventional) to be accessed serially with four or five wires. The logical protocol is based upon new instructions added to the IEEE Std 1149.1 instruction set and is the subject of this standard document. The ISC devices also contain testability circuitry compliant with IEEE Std 1149.1. This allows them to participate in manufacturing tests that detect and diagnose faults such as open solder joints and shorts either before or after they are programmed. The objective of the 1532 standard is to enhance user access to IEEE Std 1149.1 based programmable

devices. Since such devices are built on the foundation of the IEEE Std 1149.1 TAP and state machine, they already comply with that standard.

Devices that adhere to the 1532 standard realize significant savings in software, product development and manufacturing costs related to ISC. New tools not need be developed to support new vendor parts with ISC features as they arrive in the marketplace. These parts can be easily integrated immediately into existing systems and tools.

In addition, tools based on 1532 may be developed upon the existing IEEE Std 1149.1 based tool sets and knowledge base, rapidly incorporating the features of this standard. The standard also allows for multiple devices from independent vendors to be programmed concurrently. Development time and manufacturing costs (in terms of programming time) can thereby be significantly reduced, even in the multi-vendor environment. In the past, one family of devices from a single vendor may have been incompatible with another family from that same vendor, with respect to the programming process. The standard also serves board and system designers by providing a consistent framework for system initialization, stabilization and event initiation which is important for robust system design.

The standard realizes the following additional benefits:

- It serves as the development model for new entrants to the ISC industry and guarantees their immediate access to all the latest tools, development systems and test and manufacturing flows and processes.
- It facilitates innovations potentially including applications in reconfigurable computing, emulation, diagnostics and field upgrading that would be usable across vendors.
- It encourages tool vendors targeting the programmable device marketplace that there will be sufficient return on their investment to warrant tool development.

Using IEEE Std 1532 for Configuration

The standard itself is composed of two parts. The first and (now approved) portion describes the necessary hardware and functional elements that make a device 1532 compliant. The second (and currently still in development) portion describes extensions to Boundary-Scan Description Language (BSDL) that spell out how to access the functional elements in a 1532 compliant device to effect its configuration. In the next two sections, we will examine those portions of the standard.

Hardware & Functional Elements in a 1532 Compliant Device

IEEE Std 1532 is built on the foundation of IEEE Std 1149.1. This means that a 1532 compliant device must first be 1149.1 compliant. Upon that foundation, 1532 requires that the programmable device has well-defined externally observable behavior before, during and after the configuration process. To that ends the standard mandates how the configurable pins of the programmable device will behave at all times. In addition, it completely specifies when and how these pins will take on their programmed state.

System I/O pins of a 1532 compliant device have behavior summarized as follows:

- If an ISC instruction is loaded, the system I/O pins take on either HIGHZ-like or CLAMP-like behavior according to the design of the device.
- If an IEEE Std 1149.1 test mode instruction (EXTEST, INTEST, RUNBIST, CLAMP, HIGHZ) is loaded, the system I/O pins are controlled as defined by that standard.
- If an IEEE Std 1149.1 non-test mode instruction (BYPASS, IDCODE, USERCODE, SAMPLE, PRELOAD) is loaded in the instruction register, the system I/O pin behavior is determined what ISC operation was being performed when the instruction was loaded.

It is when an IEEE Std 1149.1 non-test mode instruction is loaded in an ISC device that there is a choice of behaviors for the system I/O pins. These are summarized as follows:

- If the device is in the unprogrammed state, the system I/O pins are disabled.
- If the device is performing an ISC operation, the system I/O pins take on either HIGHZ-like or CLAMP-like behavior according to the design of the device.
- If the device is in the programmed and operational state, the system I/O pins operate as determined by the programming of the device.

As illustrated in Figure 1., devices may have both ISC pins and fixed pins. In this case, the ISC pins behave as described above but the fixed pins behave according to the function of the core to which they are connected.

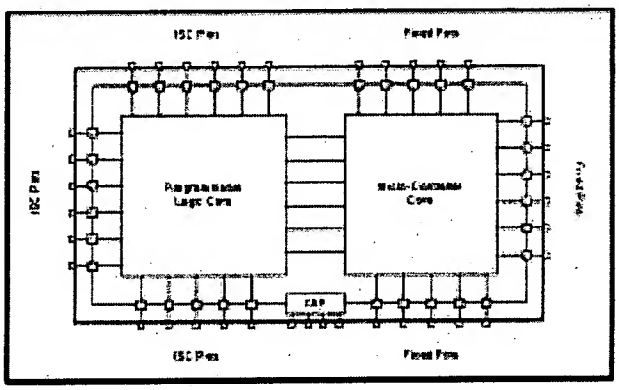

Figure 1. A 1532 compliant device with both ISC and fixed pins

The various states and transitions of a typical ISC device are all well defined by the standard. In fact, the standard defines four system modal states that correspond to the described states. In the standard they are referred to as:

Unprogrammed: In this system modal state, an ISC programmable logic device may be blank or incompletely programmed.

ISC Accessed: In this system modal state, ISC instructions may be used to read, write, verify, protect or erase the device.

ISC Complete: This system modal state exists so that an external algorithm can control transitions to the Unprogrammed or operational system modal states after ISC operations are complete.

Operational: In this system modal state, the device is ready for its "operational" mode defined by its programming.

Figure 2 illustrates these four system modal states and details the conditions for transition between them.

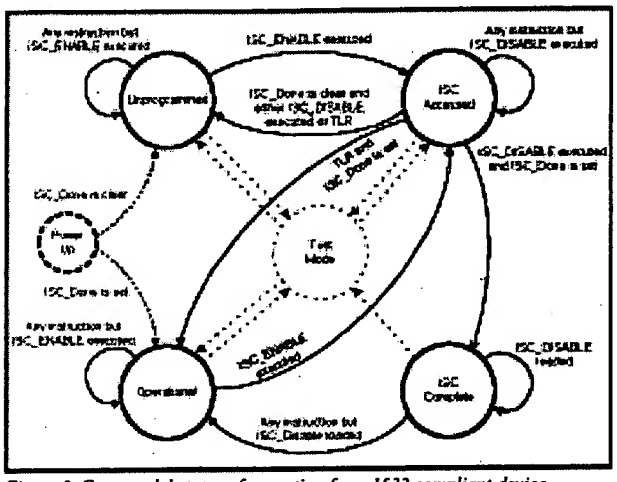

Figure 2. Four modal states of operation for a 1532 compliant device

In order to configure a device, a sequence of ISC instructions and data are loaded into the device. ISC operations are always executed when the TAP controller is transitioned to the Run-Test/Idle State. In the most elementary implementation the sequence of operations is like this:

1. Load ISC_ENABLE instruction
2. Load data associated with ISC_ENABLE instruction
3. Transition to and stay in the Run-Test/Idle TAP controller state for the prescribed amount of time.

This completes the device's preparation to begin performing ISC operations. The device's programmable system pins take on either a HIGHZ'ed or CLAMP'ed state.

4. Load the ISC_ERASE instruction
5. Load data associated with the ISC_ERASE instruction.
6. Transition to and stay in the Run-Test/Idle TAP controller state for the prescribed amount of time.

The above sequence erases the programmed contents of the device's configuration memory. Additional iterations through steps 5 and 6 may be required if the device has multiple sectors to erase.

7. Load the ISC_PROGRAM instruction
8. Load the configuration data associated with the ISC_PROGRAM instruction.
9. Transition to and stay in the Run-Test/Idle TAP controller state for the prescribed amount of time.
10. Repeats steps 8 and 9 until the device's configuration memory has been completely programmed.

The above sequence loads the programmable device's configuration memory with the designed pattern.

11. Load the ISC_DISABLE instruction
12. Transition to and stay in the Run-Test/Idle TAP controller state for the prescribed amount of time.

The above sequence marks the completion of the loading of the program pattern in the device's configuration memory. When the ISC_DISABLE instruction is displaced from the device's instruction register the device will be operational. To accomplish this the next step should be executed:

13. Load the BYPASS instruction

The device is now operational and the system pins take on their programmed behaviors.

There are variations and enhancements to the described flow that are detailed in the standard itself. The basic functionality is well described by the example. The specifics of the sequencing of instructions and data to accomplish any particular ISC operation (i.e., erase, program, verify, read, etc) are fully detailed in the BSDL extension so that applications can be developed that access all the ISC capabilities of a device algorithmically. More information about the contents of the BSDL extension now under development will be discussed in a later section.

Configuration, Test and Repair

By using 1532-compliant devices, the basic configuration of a device or chains of devices are well defined functionally, behaviorally and algorithmically. The implications here are several-fold. The external behavior of the device is fully defined by the system modal state of the device. This behavior can be fully controlled by the application and tests can be executed with full knowledge of the exact state of the device's pins. In addition, device configuration operations are fully described in the BSDL file. This makes configuration a simple matter of reading and interpreting the contents of a BSDL file in concert with the 1532 data file.

Reconfigurable System Testing

The availability of 1532 compliant programmable devices on a system board opens a rich set of possible mechanisms to incorporate full system test onto your board. This of course requires some careful planning and pre-design work [3].

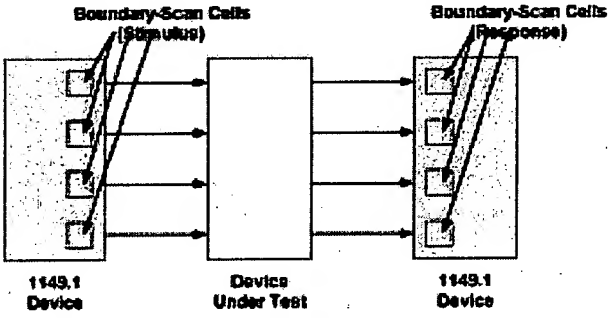

Figure 3. Using Boundary-Scan Devices to Test Other Devices

The simplest of the possible approaches for full system test is to use the boundary-scan (1149.1) capabilities of the programmable devices to assist in the static testing of other devices in the system [10,11].

The concept, illustrated in Figure 3., involves using the 1149.1 SAMPLE/PRELOAD and EXTEST instructions to control and sample system stimulus. The EXTEST instruction is used to apply stimulus to the device under test and capture results at the response side.

Stimulus can be generated and responses collected external to the system (i.e., across a network by a diagnostics host under the control of a technician at a central office) or as part of the general-purpose capabilities of the on-board microprocessor

working as a diagnostic engine. In the former case, full diagnostic and test control can be effected from the central office. Data and results are collected and distributed via the system's microprocessor connected to both the network and the 1149.1 TAP. In the latter case, the system is still network connected but the central office controls only the initiation of the diagnostic test sequence (whose actual contents are stored on board) and the system test response (which is returned to the central office). The former approach allows the potential for more operator interaction and overall test flexibility. In this case, the operator can direct tests on-the-fly as more information about the failure conditions are determined.

Note, however, that because the described method is fully controlled by the boundary scan register it allows only for gross functional or static testing. Since test vectors are shifted in serially and applied only after the shifting has been completed, at-speed problems will not be visible.

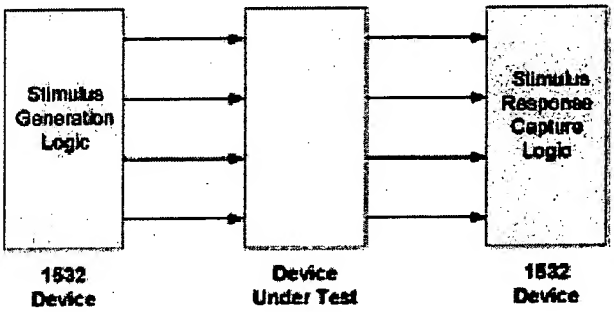

Figure 4. Using 1532 Devices to Test Other Devices

But, by leveraging programmability and the possibility of performing the programming via the simple 4 wire 1149.1 TAP, you can enhance system testability. This is accomplished as illustrated in Figure 4 by configuring devices to act as test stimulus generators and test result collectors. It is similar to taking typical single device built-in self-test (BIST) approach and breaking it into its constituent parts then reassembling those parts in a system. This is easily accomplished through the creation of such standard test constructs as multiple input shift registers, linear feedback shift registers or other signature collection and calculation registers. In addition, random or deterministic test vector generators and controlled frequency oscillators can be realized. These all represent the basic building blocks of system test. So applying the single device test and basic BIST mechanisms allow you to incorporate BIST into your system without paying additional hardware costs. The 1532 devices can be utilized for other mission functions after the BIST operations are complete. In addition, the BIST functionality can be recovered by reprogramming the 1532 devices.

The Xilinx family of FPGAs incorporates two user-available boundary-scan instructions. These can be utilized by system designers to implement special purpose boundary-scan triggered and controlled functionality[12]. These two instructions (USER1 and USER2) are quite natural building mechanisms for the operation of the system BIST functions.

The presence of the boundary-scan logic on any of the system's devices in general also provides another possible resource that can be used. For instance the SAMPLE instruction can be used to sample system signals at arbitrary times without interfering with the operation of the system BIST logic. In addition, the CLAMP and EXTEST instructions could be used and drive and hold arbitrary values on system control pins.

Reconfigurable System Repair

Once a system has been tested and diagnosed using the techniques described above the technician running the tests from the central office will have a better understanding of the problem in the system. With that understanding, the technician will have three options:

1. Still more information is needed and more tests need to be run.
2. The problem is understood to be a software issue.
3. The problem is understood to be a hardware issue.

In the case of (2) or (3), fixes can be delivered and installed remotely. In scenario (2), the system ROM is updated. In scenario (3) there are at least two further possibilities:

1. The system needs to be replaced but the exact nature of the repair is known.
2. The system can be repaired by reconfiguring some (or all) of the programmable devices in the system.

If the system has to be returned for replacement or shop repair, the technician can reconfigure the PLD's to make the system unusable. The reconfiguration could also incorporate a diagnostic code that indicates the source of the problem. Applying such a reconfiguration to the system provides a guarantee that the system will be returned promptly.

In the case where a reconfiguration of some of the system's PLD's can effect a system repair, this entails sending the configuration information to the system in a manner described in [13]. This technique described as "Internet Reconfigurable Logic" (IRL) involves designing your system such that configuration information can be received by the system and then safely installed and updated.

After the update, the same suite of tests as described previously can be applied to verify that the system is functioning as expected. This completes the process of diagnosing, repairing and verifying the system without requiring dispatch of a new system.

Conclusion

IEEE Std 1532 extends and enhances IEEE Std 1149.1 to describe fully and document the operation and external behavior of programmable logic devices. We have shown how IEEE Std 1532 can be used to configure a single programmable device and how this capability can be exploited to facilitate reconfigurable system testing and diagnosis. In addition, we have shown how 1532 compliant devices can be used to implement remote system repair.

It should be noted that the approach proposed in this paper would not be applicable to systems in which the cost of repairing the device is greater than the cost of replacing the device. It is true however; that this approach does reduce the system repair cost and therefore extends the system lifetime making the break-even point for this approach lower.

References

1. ANSI/IEEE Std 1149.1-1990 Standard Test Access Port and Boundary-Scan Architecture (includes IEEE Std 1149.1b-1994), IEEE Standards, Piscataway, N.J.
2. IEEE Std P1532 Draft Standard for In-System Configuration of Programmable Devices, IEEE Standards, Piscataway, N.J.

3. Bonnett, "Design for In-System Programming," Proceedings IEEE International Test Conference, IEEE Computer Society Press, Los Alamitos, California, 1999
4. Abramovici, Stroud, Lee and Underwood, "No-Overhead BIST for FPGAs and CPLDs," EDN, November, 1997
5. Abramovici, Chen and Stroud, "Finally, A Free Lunch: BIST (for FPGAs) Without Overhead!," Proceedings of AT&T Conference on Electronic Testing, 1995.
6. Crane, Davidson, Kane, Stroud and Wu, "Trends in Digital Device Test Methodologies," AT&T Technical Journal, Vol. 73, No. 2, 1994
7. Gibson, Gray, and Stroud, "Boundary Scan Access to BIST for Field Programmable Gate Arrays," Proceedings of the IEEE International Application Specific Integrated Circuits Conference, 1997
8. Stroud, Lee, Konala, and Abramovici, "Using ILA Testing for BIST in FPGAs," Proceedings of the IEEE International Test Conference, 1996
9. Stroud, Lee, Konala, and Abramovici, "Selecting Built-In Self-Test Configurations for Field Programmable Gate Arrays," Proceedings of the IEEE International Test Conference, 1996
10. Daniel, "Optimizing Fault Detection for Boundary-Scan Testing," ISD Magazine, September 1995.
11. Choi, Jung, "Configuration of a Boundary-Scan Chain for Optimal Testing of Clusters of Non-Boundary-Scan Devices," Proceedings of the International Conference on Computer Aided Design, 1992.
12. Virtex 2.5V Field Programmable Gate Arrays Datasheet, Xilinx, 1999
13. Jacobson, "Internet Reconfigurable Logic - Leveraging PLD's to Enhance Embedded System Functionality," Proceeding of the Embedded Internet Workshop, 1999.

Author's contact details

Neil G. Jacobson
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124 USA
Phone: 1 408 879 4885
E-mail: neil.jacobson@xilinx.com